NAME

nonamed − not a name daemon, but acts like one

SYNOPSIS

nonamed [**−qs**] [**−d**[*level*]] [**−p** *port*]

DESCRIPTION

Nonamed is not a name daemon. It can answer simple queries from **/etc/hosts**, but anything else is relayed to a real name daemon. **Nonamed** maintaines a small cache of replies it has seen from a name daemon, and will use this cache to minimize traffic if the machine is permanently connected to the Internet, or to answer requests if the machine is often disconnected from the Internet, i.e. a computer at home.

On startup **nonamed** sends a simple query to each of its name servers to see if one is up. This is repeated every five minutes" in an "at home" situation, or when necessary if the current name daemon doesn't respond. The first name server to answer is used as the current name server to answer queries.

If no name servers are found in the DHCP data or **/etc/hosts** then only the hosts file is used to answer queries, and any query for a name not in that file gets a failure response.

Nonamed accepts both UDP and TCP queries under Minix-vmd. Under standard Minix only UDP queries are accepted. 256" relayed UDP queries can be outstanding before it forgets where the first one came from.

Using the hosts file, **nonamed** can answer simple DNS queries to translate a host name to an IP address, or an IP address to a host name. Suppose **/etc/hosts** looks like this:

```
10.0.0.1        flotsam.cs.vu.nl  www
10.0.0.2        jetsam.cs.vu.nl
```

Then queries for the host names listed can be answered with the IP addresses to the left of them. An alias like "www" above is seen as a CNAME for the first host name on the line, in the same domain as the first host name if unqualified (no dots). A reverse lookup for an IP address on the left is answered by the first host name on the right. If more than one match is possible then all matches are put in the answer, so all IP addresses of multihomed hosts can be listed by multiple entries in the hosts file.

Requests for names like "flotsam.cs.vu.nl.cs.vu.nl" that are often generated on a domain search for an already fully qualified domain name are recognized and made to fail. This kludge avoids a lot of unnecessary requests to possibly unreachable name servers and client timeouts.

The name "localhost" in any domain is given the IP address 127.0.0.1.

Nonamed employs several timeouts for efficient operation:

If no UDP reply is seen in four seconds" then a new search is started for a name server in the hope of finding one that does work. A failing TCP connection will also invoke a search, the TCP connection is then made to the new name server. A client using UDP will retry eventually, a client using TCP will notice nothing but a short delay. If a TCP connection fails after 5 tries then an answer is sought in the hosts file, and failing that the connection is closed.

Any TCP operation is given five minutes" to show any action before the connection is aborted.

UDP replies from a name server are put in a cache of by default 8 (16-bit system) or 16 kilobytes (32-bit system). New queries are first sought in the cache, and if found answered from the cache. An entry in the cache is expired when the resource record with the smallest TTL (time to live) expires, unless its expire time is artificially extended by the "%stale" parameter (see below). An answer from the cache has all TTLs appropriately lowered, and the AA bit ("answer authoritive") is cleared. Any request answered by stale data is refreshed as soon as **nonamed** notices that one of the external name daemons is reachable.

Data is only cached if it is has "no error" result code, or a "no such domain" result code with a SOA record in the name server section, and all records have a nonzero TTL. The %stale parameter has no effect on the decision to cache a result.

The cache is rewritten to the cache file five minutes" after a new entry has been added. Mere changes to the order in the cache don't cause a rewrite.

Configuration through /etc/hosts

The real name servers, stale data extension, and cache size can be configured by special entries in the hosts file. For example:

```
86400         %ttl          # Answers from this file get this TTL
2419200       %stale        # Stale data may linger on for 4 weeks
32768         %memory       # 32k cache size
10.0.0.1      %nameserver   # flotsam
172.16.24.3   %nameserver   # dns1.example.com
172.16.24.6   %nameserver   # dns2.example.com

10.0.0.1      flotsam.home.example.com  www
10.0.0.2      jetsam.home.example.com
```

In this example we have two machines, flotsam and jetsam, that are at home. Answers from the hosts file get a TTL of one day, by default this is one hour". Normally there is no connection to the Internet, so any stale data in the cache is allowed to linger on for 2419200 seconds (4 weeks) before it is finally discarded. The cache size is set to 32 kilobytes. The first name server is the flotsam. On the flotsam itself this entry is ignored, but the jetsam will now run its requests through flotsam if possible. This means that both flotsam and jetsam use the cache of the flotsam. The other nameserver entries are external name servers of the Internet provider.

If no nameservers are listed in the hosts file then they are obtained from data gathered by DHCP. This is the preferred situation.

If the hosts file contains a line that says:

include *file*

Then the current hosts file is closed and the file named is read next.

Automatic calling

If your connection to the Internet is set up on demand, either in software on the machine that has the modem, or by a special box such as an ISDN router, then you need to filter the name server probes that **nonamed** sends out every five minutes" to see if a real name daemon is reachable. These probes need to be recognized as packets that must not trigger a call, and that must not keep the line up. You can either filter all IP packets destined for port 53 decimal (the **domain** port). This may be a bit too much, the first packet out is often a normal DNS request (not a probe), so you may want to do better. A probe by **nonamed** is a nonrecursive request for the name servers of the root domain. You can recognize them by looking at the flags, they are all off. Here is a typical probe in hex (twenty octets per line), followed by the names of interesting fields, and the octets values you should look for:

```
45 00 00 2D C8 19 00 00 1D 11 53 18 AC 10 66 41 AC 10 18 03
00 35 00 35 00 19 79 93 00 00 00 00 00 01 00 00 00 00 00 00
00 00 02 00 01

ip ip ip ip ip ip ip ip ip ip ip ip si si si si di di di di
sp sp dp dp xx xx xx xx id id fl fl qd qd an an ns ns ar ar
dn ty ty cl cl

45 xx xx xx xx xx xx xx xx 11 xx xx xx xx xx xx xx xx xx xx
xx xx 00 35 xx xx xx xx xx xx 00 00 xx xx xx xx xx xx xx xx
xx xx xx xx xx
```

(ip = IP header, si = source IP, di = dest IP, sp = source port, dp = dest port, id = DNS ID, fl = DNS flags, qd = query count, an = answer count, ns = nameserver count, ar = additional records count, dn = domain (""), ty = type (NS), cl = class (IN).)

So if a packet has octets 45, 11, 00 35, and 00 00 at the appropriate places then don't let it cause a call. Read the documentation of your software/router to find out how to do this. Hopefully it is possible to view the contents of the packet that triggered the last call. If so you simply let **nonamed** bring up the line once with a probe.

Remote information

The program version and name servers it is working with can be obtained with:

host −r −v −c chaos −t txt version.bind. *server*

Server is the name or IP address of the host whose name server you want to know this of. (This call is really an undocumented hack to ask the version numbers of the BIND name daemon. It just had to be implemented for **nonamed** as well.)

The % variables in the hosts file can be viewed like this:

host −r −t a %nameserver. *server*

Don't forget the dot at the end of the name. %ttl and %stale will be shown as a dotted quad, e.g. 0.36.234.0. The proper value can be computed as 36 * 65536 + 234 * 256 + 0 = 2419200.

OPTIONS

The options are only useful when debugging **nonamed**, although it can be very instructive to watch DNS queries being done.

−d[*level*]
: Set debugging level to *level* (by default **1**.) Debug mode 1 makes **nonamed** decode and display the DNS queries and replies that it receives, sends and relays. In debug mode 2 it prints tracing information about the internal jobs it executes. In debug mode 3 it core dumps when an error causes it to exit. The debugging level may also be increased by 1 at runtime by sending signal **SIGUSR1** or turned off (set to 0) with **SIGUSR2**.

[**−p** *port*]
: Port to listen on instead of the normal **domain** port.

[**−q**] Read the cache file with the debug level set to 2, causing its contents to be printed, then exit.

[**−s**] Run single: ignore hosts or cache file, only use the DHCP information. This allows another **nonamed** to be run on a different interface to serve a few programs that run there.

FILES

/etc/hosts
: Hosts to address translation table and configuration file.

/usr/run/nonamed.pid
: Process ID of the currently running **nonamed**.

/usr/adm/nonamed.cache
: Copy of the cache. Read when the program starts, written five minutes" after something has been added to it, and written when a SIGTERM signal is received, which is normally sent at system shutdown.

/usr/adm/dhcp.cache
: Data gathered by the DHCP daemon. Among lots of other junk it lists name servers that we should use.

SEE ALSO

gethostbyname(3), **resolver**(3), **hosts**(5), **inet**(8), **boot**(8), **inetd**(8), **dhcpd**(8).

RFC-1034 and **RFC-1035**.

NOTES

Do not use the %stale parameter for a PC that is directly connected to the Internet. You run the risk of getting wrong answers, a risk that is only worth taking for a system that is mostly disconnected from the Internet.

You can specify one or more remote name servers in **/etc/resolv.conf** so that nonamed isn't needed. This will save memory, but you'll lose **nonamed**'s cache and its "offline" tricks. That's no problem if you can use a neighbouring name daemon on another PC at home.

The default cache size seems to be more than enough for normal use, but if you do decide to make it larger then don't forget to increase the stack size of the program under standard Minix.

Don't let two **nonamed**'s forward queries to each other. They will pingpong a query over the network as fast as they can.

BUGS

The idea of serving "stale DNS data" will probably make some purists violently sick...

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)